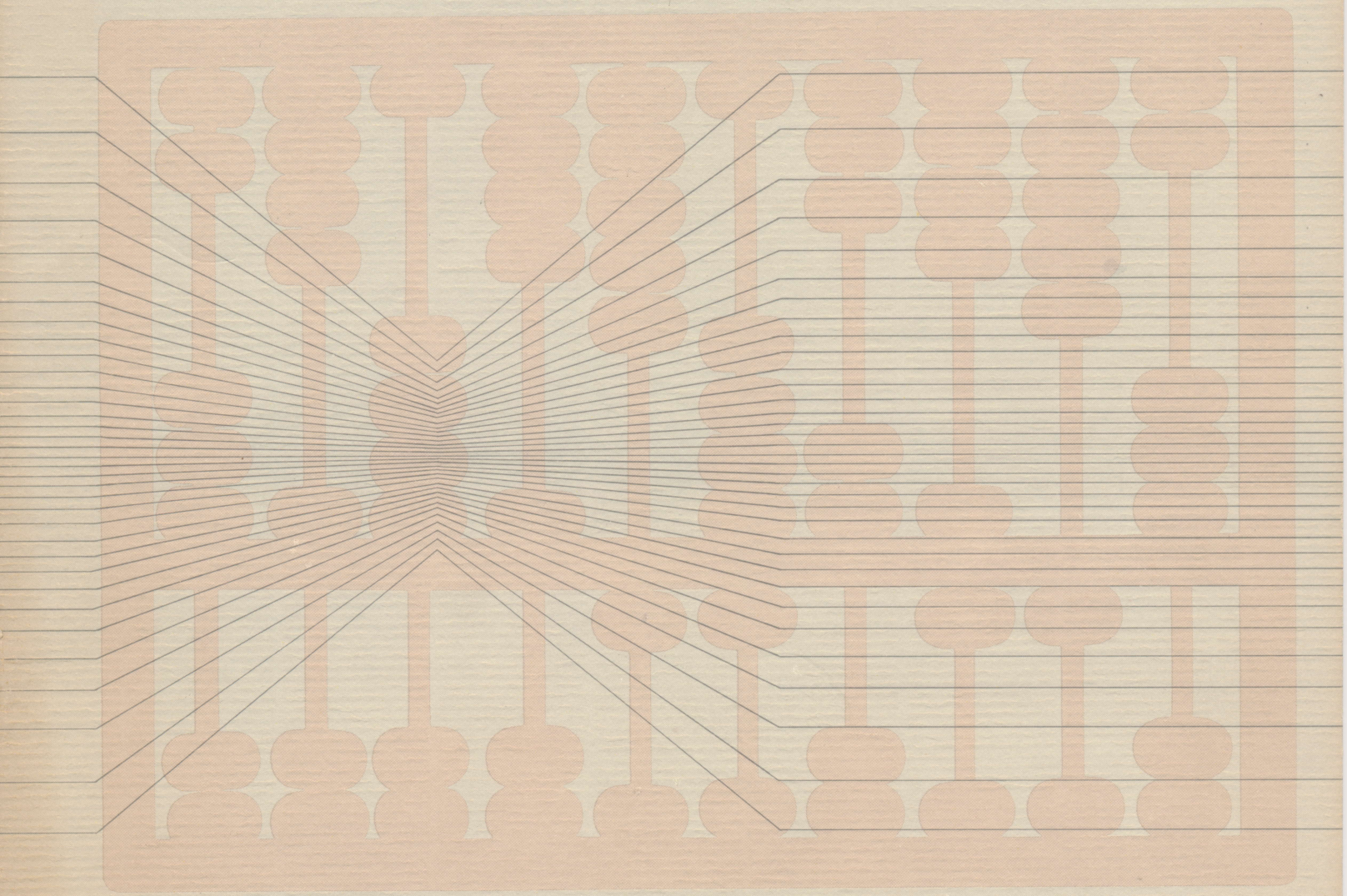
I. P. Sharp Associates

RETRIEVE



RETRIEVE

A SHARP APL

DATA BASE RETRIEVAL SYSTEM

Copyright April 1979.

All rights reserved. This book or parts thereof may not be reproduced by any means without the written permission of I.P. Sharp Associates Limited.

Printed in Canada.
Publication Code: 79RE04

CONTENTS

	Page
Introduction	1
Getting Started	2
Brief Function Description	3
Access Parameters	
View	5
Fact	5
Object	5
Frequency	6
Detailed Function Description	7
Examples	14
Error Messages	19
Programming Notes	19

INTRODUCTION

The workspace 55 *RETRIEVE* contains a set of functions which enable you to retrieve data in standard format from a number of the public data bases available on the SHARP APL system. RETRIEVE is **not** a generalized data base management system, but rather provides a retrieval mechanism which allows you to access the data contained in these data bases. Typically, RETRIEVE is used in applications which require data for further processing such as portfolio management and time series analysis.

The retrieval mechanism for this system is designed so as to be compatible with a wide variety of data bases. This means that even if your application requires data from several data bases, you need only acquaint yourself with one method of retrieval. In addition, shadowing problems that occur while using retrieval functions with the same name are eliminated.

The RETRIEVE system refers to a data base as a **view**. Each data base, or view, consists of a collection of logically-related **facts** which are stored for one or more **objects**. For example, the view called *'TSEINDEXHIST'* contains data associated with the Toronto Stock Exchange Indices. Within this view, facts such as index value and dividend yield are stored for objects such as the gold index, the composite index, and so on.

Most of the data available using RETRIEVE is in the form of time series (i.e., a series of values or observations associated with a particular time interval or frequency). Facts are available with multiple frequencies. For example, a fact such as the volume of a stock trading on an exchange, can exist as a daily, weekly and/or monthly series. The latter two are summarized from the daily series by summing the volume by week and month, respectively.

In addition, static facts such as names and descriptions associated with time series are also accessible using RETRIEVE.

A description of each of the views currently available using RETRIEVE is provided in the RETRIEVE View Directory which can be obtained from your local I.P. Sharp Associates representative. The View Directory is also available on-line and can be accessed using the function *∆VIEWDOC* (see the section in this manual entitled "Detailed Function Description").

This manual assumes that you have some knowledge of the APL language and the SHARP APL time-sharing service. An introductory manual on SHARP APL can be obtained by contacting your local Sharp representative.

GETTING STARTED

To access the functions contained in the RETRIEVE workspace, enter

```
)LOAD 55 RETRIEVE
```

In order to retrieve data from a view, you have to provide the system with the four items of information outlined below. These items, called parameters, make it possible for RETRIEVE to uniquely identify any data item within a view.

1. To indicate which view contains the data you wish to access, you first specify a **view name**.

2. To indicate which facts within this view you wish to retrieve, you specify one or more **fact identifiers**.

3. If the fact, or facts, you specify are stored for more than one object, you have to qualify your retrieval request with an **object identifier**.

4. If the facts being retrieved are time series, you qualify your retrieval request by specifying a **timeframe** of interest.

The view name, as well as fact and object identifiers, can be obtained from the RETRIEVE View Directory. Prior to performing an access, you should use the Directory to obtain the necessary parameter information for the retrieval in question.

For example, to retrieve the index value, dividend yield and name as of December 30, 1976 for the Toronto Stock Exchange Composite Index, first familiarize yourself with the parameters necessary to identify the required data. After identifying the view name `'TSEINDEXHIST'`, fact mnemonics `'TSE1'`, `'TSE4'` and `'TSE11'`, the object mnemonic of `'TXTT'` and the date format, you can then perform an access by passing these parameters to the RETRIEVE functions as shown below.

```
\VIEW 'TSEINDEXHIST'                    - specify the view name.

DAILY' ΔTIME 19761230                   - establish a timeframe by specifying a frequency and date.

?←'TXTT' ΔGET 'TSE1,TSE4,TSE11'  - request that facts 1, 4 and 11 be returned for the object
                                          identified by the mnemonic 'TXTT' (i.e., the composite
                                          index).
```

Detailed descriptions of the functions used in the above example can be found in the following sections.

Views accessible by RETRIEVE are defined by the data base administrator. The available data is supplied to I.P. Sharp Associates from various sources. To gain access to some views, it may be necessary to enter into a contractual arrangement with the source supplier.

BRIEF FUNCTION DESCRIPTION

There are three types of functions within RETRIEVE: retrieval functions, state setting functions and special purpose functions. Retrieval functions are used to retrieve data from the various data bases which are accessible via the RETRIEVE system. State setting functions assist in forming access parameters and in controlling the result from *ΔGET*. Special purpose functions are used to summarize time series data, to restart suspended functions and to obtain on-line documentation.

To perform an access using RETRIEVE, first define your view by passing the view name of interest to the function *ΔVIEW*. If you are accessing a time series, the frequency and timeframe must be specified by passing the frequency and dates of interest to the function *ΔTIME*. You can then access the data of interest by passing the fact identifier and an optional object identifier to the function *ΔGET*.

A description of the parameters used with *ΔVIEW*, *ΔTIME* and *ΔGET* can be found in the section entitled "Access Parameters". A list and brief description of all RETRIEVE functions appears below. For a more detailed description of these functions, refer to the section entitled "Detailed Function Description".

Access Functions

ΔVIEW - defines the logical view from which data is to be retrieved.

ΔTIME - defines a frequency and timeframe for which time series data is to be retrieved.

ΔGET - retrieves data for the specified facts and objects.

ΔTO - used in conjunction with *ΔTIME* to define a range of dates.

ΔSEARCH - allows you to search the list of available facts, objects and dates within a view.

State Setting Functions

ΔCLEAR - returns all state settings to their default.

ΔRESULTTYPE - allows you to specify whether the data accessed with *ΔGET* is to be returned in variables within an explicitly-returned package, or in global variables within the active workspace.

ΔRENAME - allows you to specify the names of the variables returned by *ΔGET*.

ΔPREFIX - defines a prefix for all object mnemonics passed as a left argument to *ΔGET*.

ΔERRORTYPE - allows you to specify that error messages are to be returned as an explicit or as a non-explicit result.

ΔFNSRES - allows you to specify whether the file functions called by RETRIEVE are to be workspace resident or file resident.

ΔSTATE - displays current state settings, frequency and view name.

Special Purpose Functions

ΔSUMMARIZE - summarizes time series according to a specified frequency and summarization method.

ΔRESTART - returns the restart point after function suspension.

ΔVIEWDOC - allows you to generate a list of current view names and/or a description of each available view.

ACCESS PARAMETERS

When performing an access you specify view, fact and object. When accessing time series, you must also specify a frequency and timeframe. A description of each of these parameters follows.

1. View Parameter

A view is uniquely identified by its name as defined in the View Directory. This name is a character vector of variable length. The view name, enclosed in quotes, is passed as a right argument to the function *ΔVIEW*. For example, the view name, *'TSEINDEXHIST'*, identifies the view containing data associated with the Toronto Stock Exchange Indices.

2. Fact Parameter

A fact within a view is uniquely identified by a character mnemonic of variable length, or by its numeric identifier, both of which are defined in the View Directory. The fact mnemonic, enclosed in quotes, (or its numeric identifier) is passed as a right argument to the function *ΔGET*. Fact mnemonics are made up of the first three characters of the view name suffixed by the fact number. Examples of fact mnemonics for the view *'TSEINDEXHIST'* are *'TSE1'*, *'TSE4'* and *'TSE11'* which identify the index value, dividend yield and name, respectively, for a given index. The corresponding numeric identifiers for these facts are 1, 4 and 11.

3. Object Parameter

When retrieving a fact which is stored for more than one object in a view, it is necessary to supply an additional object identifier which points to the particular occurrence of interest. The object parameter uniquely identifies the object for which the specified facts are returned.

An object mnemonic, enclosed in quotes, (or its corresponding numeric identifier) is passed as a left argument to the function *ΔGET*. Examples of object mnemonics for the view *'TSEINDEXHIST'* are *'TXTT'* and *'TXGL'* which identify the Toronto Stock Exchange Composite Index and the Gold Index, respectively. The equivalent numeric identifiers for these object mnemonics are 62 and 6, respectively. Object mnemonics and their corresponding numeric identifiers are defined in the View Directory.

There are two reserved words, *'DITTO'* and *'FDESC'*, which can be passed to *ΔGET* as fact or object mnemonics.

If you are performing successive retrieves in which one, or both, of the arguments to *ΔGET* do not change, you can specify *'DITTO'* instead of repeating the argument list. This is particularly useful when respecifying a long list of fact or object identifiers.

The reserved word, *'FDESC'*, is used to retrieve fact descriptions. In this case, *'FDESC'* is passed as a right argument to *ΔGET*. The identifiers of the facts to be described are passed as a left argument to *ΔGET* (see example 5 in the section entitled "Examples").

4. Frequency Parameter

When accessing time series data, it is necessary to specify a frequency and dates in order to establish the timeframe for which the data is to be retrieved. The frequencies and timeframes available for each view are defined in the View Directory.

Frequency is specified by a character mnemonic such as *'DATED'*, or *'DAILY'* (see table below) which is passed as a left argument to the function *ΔTIME*. It is only necessary to specify the first three characters of each frequency mnemonic. An elided left argument to *ΔTIME* signifies the default frequency *'DATED'*. The frequency *'DATED'* is used with series that have an irregular frequency.

Dates are specified as numerics in the form of year, month and day (YYYYMMDD) and are passed as a right argument to the function *ΔTIME*. Dates are the index into the time series being retrieved by *ΔGET*. If you specify a date beyond the range of the series being accessed, *ΔGET* returns an error message.

Depending on the specified frequency, part of the date can be elided. The frequencies *'DATED'*, *'DAILY'*, and *'WEEKLY'* require a full date consisting of year, month and day (YYYYMMDD) to identify a period. For frequencies *'MONTHLY'*, *'QUARTERLY'* and *'SANNUAL'*, you need only specify year and month (YYYYMM) to identify a period. With a frequency of *'ANNUAL'*, only the year (YYYY) needs to be specified.

Although you need only specify the minimum date to identify a period within a frequency, the function *ΔTIME* also accepts dates in the full form of YYYYMMDD.

The following is a list of the available frequency mnemonics and the corresponding minimum date format required to identify a single period:

Frequency	Date format
DATED	YYYYMMDD
DAILY	YYYYMMDD
WEEKLY	YYYYMMDD
MONTHLY	YYYYMM
QUARTERLY	YYYYMM
SANNUAL	YYYYMM
ANNUAL	YYYY

It is also valid to specify dates with less than the minimum number of parts necessary to identify a single period. This has the effect of generating a range of dates. For example, if you specify a frequency of *'WEEKLY'* and a date of 197801, dates for all four weeks in January 1978 are generated. If you specify a frequency of *'WEEKLY'* and a date of 1978, dates for all fifty-two weeks in 1978 are generated. You can also specify a range of dates by using the function *ΔTO* in conjunction with *ΔTIME* (see the section entitled "Detailed Function Description").

DETAILED FUNCTION DESCRIPTION

RETRIEVE functions return explicit results for valid parameters. Invalid parameters have two effects, depending upon the state setting you specify with *ΔERRORTYPE*.

If *ΔERRORTYPE* has the default setting of *'NONEXPLICIT'*, and you attempt to execute a function with invalid parameters, the system displays an error message and no result is returned. This could also result in a *VALUE ERROR* if the function is called in a line of code which expects an explicit result. When this happens, you should correct the incorrect parameter and re-execute the statement.

If the error state is set to *'EXPLICIT'* and a RETRIEVE function is executed with invalid parameters, the error message is returned as an explicit result. You are expected to test for this condition in the calling function and handle it accordingly (see example 8 in the section entitled "Examples").

The functions that constitute the RETRIEVE system are described in detail below.

Access Functions

Syntax: *R*←*ΔVIEW* **viewname**

Description: Defines the view from which data is to be retrieved.

Parameters: The right argument must be a character view name enclosed in quotes.

Result: If the view name is valid, *R* contains an empty character array (`0 0⍴ ''`). An invalid view name (or other error) results in a variable length character error message being returned.

Syntax: *R*←[**frequency**] *ΔTIME* **dates**

Description: Defines the frequency and timeframe used to select data items from a time series.

Parameters: The left argument must be one of *'DATED'*, *'DAILY'*, *'WEEKLY'*, *'MONTHLY'*, *'QUARTERLY'*, *'SANNUAL'*, or *'ANNUAL'*. Only the first three characters of the frequency mnemonics need be specified. If the left argument is not specified, the default is *'DATED'*.

The right argument must be a scalar or vector of numeric dates for which data items are to be accessed. The dates must be in the form described in the previous section.

Result: If the frequency mnemonic and date format are valid, *R* is returned as an empty character array. Otherwise, a variable length character error message is returned.

Syntax: *R*←[**object identifier**] Δ*GET* **fact identifier**

Description: Retrieves (gets) data items corresponding to the specified fact and qualified by the optional object for the previously specified view and frequency. The data for each fact is returned either explicitly (in a package) or as a global variable in the active workspace.

Parameters: The right argument must be a valid fact mnemonic (enclosed in quotes) or its corresponding numeric identifier. The left argument must be a valid object mnemonic (enclosed in quotes) or its corresponding numeric identifier. The left argument points to an object within the current view for which the specified facts are returned.

When fact or object mnemonics are passed to Δ*GET* in the form of vectors, they must be separated by commas. If they are passed to Δ*GET* in matrix form, each row must be a separate mnemonic.

Result: For valid parameters, Δ*GET* returns variables containing arrays of data. Each fact specified as a right argument to Δ*GET* is returned in a separate variable. Depending on the setting specified with the function Δ*RESULTTYPE*, these variables are returned from Δ*GET* either in an explicitly returned package or as global variables in the active workspace. Incorrect parameters result in an error message.

The default setting for Δ*RESULTTYPE* is '*PACKAGE*' which causes Δ*GET* to return a package of variables containing the required data as an explicit result. The package can be manipulated using SHARP APL package functions. For further details regarding packages, refer to SATN 14.

When Δ*RESULTTYPE* is set to '*GLOBAL*', the function Δ*GET* defines the variables of data as globals in the active workspace and returns an empty array (`0 0⍴ ''`) as an explicit result. You must ensure that name conflict between objects present in the workspace and the variables defined by Δ*GET* does not occur.

The names of variables containing the arrays of data are derived from the mnemonic or numeric identifiers specified for the fact. A fact specified by a mnemonic is returned in a variable of the same name. For example, the fact specified by the mnemonic '*TSE*1', is returned in a variable called *TSE*1.

When numeric identifiers are used, the variable name is made up of the first three characters of the view name suffixed with the numeric identifier for the fact. For example, if you retrieve facts 1 and 2 from the view '*TSEINDEXHIST*', the results can be found in the variables *TSE*1 and *TSE*2.

You can override the names of the variables resulting from Δ*GET* by using the state setting function Δ*RENAME* (described below).

When you access a time series, the first row of the fact array contains the dates for the series in successive rows (i.e., the number of columns in the matrix corresponds to the number of periods (dates) specified for the Δ*GET*). The number of successive rows is equal to the number of objects specified in the left argument to Δ*GET*. If the left argument is not specified, the array contains only one successive row.

When static facts such as names or descriptions are accessed, the number of rows in the fact array correspond exactly to the number of objects passed as a left argument to *ΔGET*. If no object is specified, a single row fact array is returned. The number of columns in the array corresponds to the length of the data item being accessed.

Syntax: *R*←**start date** *ΔTO* **end date**

Description: This function is used in conjunction with *ΔTIME* and *ΔSEARCH* to specify a timeframe between the start date and the end date within a specified frequency.

Parameters: Single dates must be specified in the left and right arguments. The dates must be in the form described in the subsection entitled "Access Parameters". The start date must be earlier than the end date.

Result: A two-element vector of the user-supplied dates negated. The functions *ΔTIME* and *ΔSEARCH* interpret negative dates as a range specification.

Syntax: *R*←**type** *ΔSEARCH* **mask**

Description: Searches the list of objects, facts and dates within the current view according to the specifications given with the mask.

Parameters: The left argument must be a valid mnemonic enclosed in quotes which defines the list to be searched (i.e., *'OBJECT'*, *'FACT'* or any frequency mnemonic such as *'DAILY'*). Only the first three characters of each mnemonic need be specified.

The right argument, **mask**, must be either character (in which case it must be enclosed in quotes) or numeric as follows:

1. If **mask** is an empty character vector (`''`), and **type** is either *'FACT'* or *'OBJ'*, all fact or object mnemonics within the current view are returned in *R*.

2. If **mask** is a character string, and **type** is either *'FACT'* or *'OBJ'*, the result is all fact or object mnemonics that start with the specified character string.

3. If **mask** is a character string ending with a blank, and **type** is either *'FACT'* or *'OBJ'*, all fact or object mnemonics that **exactly** match the characters preceding the blank are returned.

4. When searching for compound object mnemonics, parts of the mnemonic can be omitted by substituting an elision character for the elided part of the mnemonic. Within each part of the supplied mnemonic, the search is done according to rules 2 and 3 above.

5. If **mask** is an empty numeric vector (`⍳0`) and **type** is either *'FACT'* or *'OBJ'*, all numeric fact or object identifiers are returned.

If **mask** is an empty numeric vector (⍳0) and **type** is a frequency mnemonic (i.e., *'MONTHLY'*) all valid dates for the specified frequency within the current view are returned.

6. If **mask** is a single numeric identifier or date, and **type** is either *'FACT'*, *'OBJ'*, or a frequency mnemonic the same is returned if it exists. Otherwise, an empty numeric array is returned.

7. If **type** is a valid frequency mnemonic and **mask** is specified using *ΔTO*, the result is all valid dates between the start and end dates passed to *ΔTO*.

Note: Character **masks** (see rules 1, 2, 3 and 4 above) are invalid when **type** is a frequency mnemonic.

Result: If **mask** is character and the search is successful, the result is a character matrix with one object (or fact) mnemonic per row.

If **mask** is numeric and the search is successful, the result is a numeric vector.

If the search is unsuccessful, the result is an empty numeric/character array or an error message.

State Setting Functions

Syntax: *ΔCLEAR*

Description: Returns all state settings to their default conditions.

Parameters: None

Result: None

Syntax: *ΔRESULTTYPE* **type**

Description: Allows you to specify whether the result from *ΔGET* is returned in variables within an explicitly returned package (default setting) or in global variables within the active workspace.

Parameters: The right argument must be one of the valid type mnemonics *'PACKAGE'* or *'GLOBAL'*. Only the first three characters need be specified.

Results: If *'PACKAGE'* (the default) is specified, the function *ΔGET* returns all variables containing data in an explicitly-returned package.

If *'GLOBAL'* is specified, *ΔGET* defines all the variables containing data as globals in the active workspace and returns an empty character array (0 0⍴ '') as an explicit result.

Syntax: *R*←**variable names to be overridden** *ΔRENAME* **new variable names**

Description: Allows you to specify new names for the variables returned by *ΔGET*. *ΔRENAME* must be invoked prior to *ΔGET*.

Parameters: The left argument must be a character vector (enclosed in quotes) containing the variable names to be overridden, separated by commas. The right argument must be a character vector (enclosed in quotes) containing the new variable names, separated by commas.

Result: If the number of names specified in the left and right argument are not equal or consist of invalid characters, an error message results; otherwise an empty character array is returned. The next time *ΔGET* is invoked with facts corresponding to those passed as the left argument of *ΔRENAME*, data for those facts will be returned in variables with the names as specified in the right argument of *ΔRENAME*. Names not referenced by *ΔGET* are ignored.

Syntax: *ΔPREFIX* **character prefix**

Description: Prefixes all object mnemonics passed as a left argument to *ΔGET* with the prefix specified in the right argument. Specifying an empty character array (`''`) switches off the prefix facility. This function must be invoked prior to *ΔGET*.

Parameters: Right argument must be character and enclosed in quotes.

Result: Empty character array or an error message.

Syntax: *ΔERRORTYPE* **type**

Description: Allows you to specify whether error messages are to be returned as an explicit or non-explicit result by RETRIEVE functions.

Parameters: The right argument must be either *'EXPLICIT'* or *'NONEXPLICIT'*. Only the first three characters need be specified.

Result: If *'NONEXPLICIT'* (default setting) is specified, RETRIEVE displays all error messages (i.e., does not return them as explicit results).

If *'EXPLICIT'* is specified, all error messages are returned as explicit results. You are expected to test for error conditions and handle them accordingly.

Syntax: *ΔFNSRES* **type**

Description: Allows you to specify whether or not the file functions called by RETRIEVE are to remain in the active workspace after they have been executed.

Parameters: The right argument must be one of *'WS'* or *'FILE'*.

Result: If *'WS'* (default setting) is specified, the RETRIEVE file functions are left in the workspace after they've been fetched from file and executed. This means that file functions are fetched only once for each view.

If *'FILE'* is specified, the file functions are erased immediately after they've been executed, thereby not occupying space in the active workspace.

To guarantee that you are using the latest software at all times, *ΔVIEW* erases all file functions present in the workspace each time it is executed.

Syntax: *ΔSTATE*

Description: Displays current state settings, frequency and view name.

Parameters: None

Result: A character display of current state settings.

Special Purpose Functions

Syntax: *R*←**frqsum** *ΔSUMMARIZE* **data**

Description: Summarizes time series according to the specified frequency and summarization method.

Parameters: The right argument must be a matrix of data. The first row contains dates in the form YYYYMMDD and successive rows contain the time series to be summarized. Typically, this is a variable (returned from *ΔGET*) containing a fact array which you want to summarize.

The left argument must be a two-element vector in which the first element is the frequency code and the second element is the summarization code. The frequency determines the frequency of the series to be returned, and must be less than that of the original series. The summarization code determines the method used to calculate the new series.

The following is a list of valid frequency and summarization codes and their corresponding descriptions.

Frequency Code	Frequency	Summarization Code	Method
1	weekly	1	Totals
2	monthly	2	Averages
3	quarterly	3	First Value
4	semi-annual	4	Last Value
5	annual	5	Minimum Value
		6	Maximum Value

Result: A matrix consisting of a summarized series corresponding to the matrix you specified as a right argument to *ΔSUMMARIZE*. The result usually has less columns (periods) than the matrix specified as the right argument. The dates in row one correspond to the period end dates.

Syntax: *R←ΔRESTART*

Description: Returns as a result the restart point in whatever function is suspended after tying all RETRIEVE files that were tied prior to the suspension.

Parameters: None

Result: Restart point or an error message.

Syntax: *R←ΔVIEWDOC* **viewname**

Description: Allows you to generate a list of current view names and/or a description of each available view.

Parameters: The right argument must be an empty character vector or a valid view name enclosed in quotes.

Result: If the right argument is an empty character vector, *R* contains an *N*×18 array of valid view names.

If the right argument is a valid view name, *R* contains a variable length character description for the specified view.

An invalid view name (or other error) results in a variable length error message.

EXAMPLES

The following examples are based on the view *'TSEINDEXHIST'* which contains historical data for the sixty-two indices that constitute the Toronto Stock Exchange Indices.

1. To retrieve the daily index value for the composite index for the year 1976, specify the view name, establish a timeframe and identify the object mnemonic for the composite index and fact mnemonic for the index value as follows:

```
ΔVIEW 'TSEINDEXHIST'

'DAILY' ΔTIME 19760101 ΔTO 19761230

DATA←'TXTT' ΔGET 'TSE1'
```

The above results in the package, *DATA*, containing the variable *TSE1*. This variable has two rows, the first containing a series of dates and the second row containing the daily index values for the *'TSE300'* composite index. The 253 columns of the variable *TSE1* correspond to all the trading days in 1976.

It is also valid to identify the object and fact by their numeric codes, 62 and 1, respectively. In addition, the range of dates can be specified without using *ΔTO* by passing an elided date to *ΔTIME* in the form YYYY as shown below. The result is identical to the previous example.

```
ΔVIEW 'TSEINDEXHIST'

'DAILY' ΔTIME 1976

DATA←62 ΔGET 1
```

2. Suppose after performing the retrieval shown above, you decide to retrieve the dividend yield, price earning ratio, and name for the composite index. To do this, first identify the necessary fact mnemonics for the facts to be retrieved and pass them to *ΔGET*. The view and timeframe are still in effect from the previous access.

```
DATA←'TXTT' ΔGET 'TSE4,TSE5,TSE11'
```

The resulting package contains three variables, namely *TSE4*, *TSE5* and *TSE11*. The variables *TSE4* and *TSE5* each have two rows, the first containing dates and the second the series being accessed. The number of columns in both variables correspond to the 253 trading days in 1976. The name/description of the composite index is contained in the 1x43 character matrix in the variable *TSE11*.

Again, numeric object and fact identifiers can be substituted for the character mnemonics as shown below.

```
DATA←62 ΔGET 4 5 11
```

3. To retrieve the monthly index value, divided yield and name/description for the Insurance and Bank Indices, you need to specify the object mnemonics *'TXBNK'* and *'TXINS'*. If you first pass the character string *'TX'* to the function *ΔPREFIX*, you need only specify the object mnemonics *'BNK'* and *'INS'* as shown below.

```
'MONTHLY' ΔTIME 1976

ΔPREFIX 'TX'

DATA←'BNK,INS' ΔGET 1 4 11
```

In this case, the variables *TSE*1 and *TSE*4 have 3 rows and 12 columns. The first row contains dates and the second and third rows contain the series for the Insurance and Bank Indices. The columns correspond to the 12 months of the year. *TSE*11 holds name/descriptions for *'TXBNK'* and *'TXINS'*.

4. Suppose you wish to retrieve the monthly index value, dividend yield and name for the Insurance and Bank Indices and have the results returned in the variables *INDEX*, *DIV* and *NAME* instead of *TSE*1, *TSE*4 and *TSE*11. Use the function *ΔRENAME* as shown below. To do this, enter

```
'MONTHLY' ΔTIME 1976

ΔPREFIX 'TX'

'TSE1,TSE4,TSE11' ΔRENAME 'INDEX,DIV,NAME'

DATA←'BNK,INS' ΔGET 1 4 11
```

Because an access has already been performed for the objects *'TXBNK'* and *'TXINS'*, it is not necessary to respecify the object. Instead, the reserved word *'DITTO'* can be substituted for the object identifier as follows:

```
'MONTHLY' ΔTIME 1976

ΔPREFIX ''

'TSE1,TSE4,TSE11' ΔRENAME 'INDEX,DIV,NAME'

DATA←'DITTO' ΔGET 1 4 11
```

The use of the reserved word *'DITTO'* is particularly useful in respecifying long lists of fact or object identifiers.

5. To retrieve fact descriptions, the numeric or character fact identifiers are passed as a left argument to *ΔGET* and the reserved word *'FDESC'* is specified as a right argument. The fact descriptions are returned in the variable *FDESC* within the explicitly-returned package from *ΔGET*.

```
DATA←1 2 4 ΔGET 'FDESC'
```

6. It may be more convenient to have *ΔGET* return a global variable instead of a package if you were retrieving the fact descriptions to perform a visual check. Specifying *'GLOBAL'* to the function *ΔRESULTTYPE* causes *ΔGET* to return all results as global variables.

```
      ΔRESULTTYPE 'GLOBAL'
      ρ1 2 4 ΔGET 'FDESC'
0 0
      ρFDESC
3 40
      FDESC
INDEX VALUE
RELATIVE STRENGTH TO GROUP
DIVIDEND YIELD (IN PERCENT)
```

7. The following examples show how *ΔSEARCH* can be used to search object, fact and frequency lists.

```
      ΔVIEW 'TSEINDEXHIST'
      'OBJECT' ΔSEARCH 'TXT'
TXTOB
TXTRN
TXTR
TXTRP
TXTUT
TXTST
TXTT

      'OBJECT' ΔSEARCH 'TXTT'
TXTT

      'OBJECT' ΔSEARCH ''
TXMM
TXINM
TXMET
.
.
.

      'OBJECT' ΔSEARCH ι0
1 2 3 4 5 6 7 8 9 10 11 . . .

      'OBJECT' ΔSEARCH 62
62

      'FACT' ΔSEARCH 'TSE12'
TSE12

      'DAILY' ΔSEARCH ι0
19760102 19760105 19760106 19760107 . . .

      'DAILY' ΔSEARCH 19760102 ΔTO 19760106
19760102 19760105 19760106
```

8. The function *ΔERRORTYPE* allows you to specify whether error messages are to be returned as explicit or non-explicit results as shown in the following example.

```
      R←'DAILY' ΔSEARCH ⍳0
ER:11 VIEW NOT SPECIFIED
VALUE ERROR
      R←'DAILY' ΔSEARCH ⍳0
                ^
      ΔERRORTYPE 'EXPLICIT'
      R←'DAILY' ΔSEARCH ⍳0
      ⍴R
24
      R
ER:11 VIEW NOT SPECIFIED
```

9. The function *ΔFNSRES* allows you to specify whether the file functions called by RETRIEVE are to be file or workspace resident.

```
      ΔFNSRES 'WS'

      ΔFNSRES 'FILE'
```

10. The function *ΔSTATE* displays current state settings, frequency and view name.

```
      ΔSTATE
PREFIX NOT SET
RENAME NOT SET
FUNCTIONS WS   RESIDENT
ERROR MESSAGES EXPLICIT
RESULT TYPE GLOBAL
FREQUENCY DAILY
VIEW TSEINDEXHIST
```

11. The function *ΔCLEAR* returns state settings to their defaults.

```
      ΔCLEAR
      ΔSTATE
PREFIX NOT SET
RENAME NOT SET
FUNCTIONS WS   RESIDENT
ERROR MESSAGES NONEXPLICIT
RESULT TYPE PACKAGE
FREQUENCY DAILY
VIEW TSEINDEXHIST
```

Note that frequency and view name are not affected by *ΔCLEAR*.

12. Using *ΔSUMMARIZE* you can summarize some previously accessed fact array into a lower frequency.

    Suppose you wanted to generate a monthly series for index highs from a previously accessed daily fact array. To achieve this, pass the daily fact array as a right argument to *ΔSUMMARIZE* and the appropriate frequency and summarization code as a left argument, as shown below.

```
      MONTHLYHIGH←2 6 ΔSUMMARIZE TSE1
```

13. The function *ΔRESTART* is used to restart execution after function suspension, as shown below.

```
      )SI
ΔTIME[6] *
      →ΔRESTART
```

14. Using *ΔVIEWDOC* you can generate a list of current view names and get a complete description of a specified view, as shown below.

```
      ΔVIEWDOC ''
TSEINDEXHIST
OPTIONCDN
COMMODITIES


      ⍴DOC←ΔVIEWDOC 'TSEINDEXHIST'
10161
```

    The variable *DOC* contains the name, view number, and a complete description for the view *TSEINDEXHIST*, and provides all fact and object identifiers for this view.

ERROR MESSAGES

Error messages generated by the RETRIEVE system are variable-length character vectors. The first five characters of all error messages consist of *'ER:'* followed by a two-digit error number.

For example:

```
ER:01 FILE SUBSYSTEM NOT AVAILABLE
```

PROGRAMMING NOTES

The functions in the RETRIEVE workspace reference functions which reside on file. These file resident functions are brought into the active workspace as required. The state setting specified with *ΔFNSRES* determines whether or not the file functions remain in the workspace after they have been executed. The names of all file resident functions start with delta (Δ) and the first two characters of each name are underscored (e.g., *Δ_GET*). This naming convention allows the functions associated with RETRIEVE to be recognized. To avoid name conflict, do not define any objects with names used by RETRIEVE.

When you define a view using the function *ΔVIEW*, a number of system variables are defined in the workspace as globals. The names of these variables start with delta (Δ) and are underscored. Again, you should avoid name conflict by not defining objects with names used by global system variables.